Table des matières

[1 Présentation 3](#_Toc124344774)

[1.1 Description générale 3](#_Toc124344775)

[1.2 Règles du jeu 3](#_Toc124344776)

[1.3 Cinématique des écrans 4](#_Toc124344777)

[2 Conception – Diagramme de classe 5](#_Toc124344778)

[2.1 Présentation générale 5](#_Toc124344779)

[2.2 Présentation détaillée Game1 7](#_Toc124344780)

[2.3 Présentation détaillé Menu 9](#_Toc124344781)

[2.4 Présentation détaillé ChoixNiveau 10](#_Toc124344782)

[2.5 Présentation détaillé Regle 10](#_Toc124344783)

[2.6 Présentation détaillé Desert 11](#_Toc124344784)

[2.7 Présentation détaillé Snow 11](#_Toc124344785)

[2.8 Présentation détaillé GameOver 11](#_Toc124344786)

[2.9 Présentation détaillé Win 11](#_Toc124344787)

[2.10 Présentation détaillé Pingouin 11](#_Toc124344788)

[2.11 Présentation détaillé Snowball 11](#_Toc124344789)

[2.12 Présentation détaillé MonstreVolant 11](#_Toc124344790)

[2.13 Présentation détaillé MonstreRampant 11](#_Toc124344791)

[2.14 Présentation détaillé Trap 11](#_Toc124344792)

[2.15 Présentation détaillé Recompenses 11](#_Toc124344793)

[2.16 Présentation détaillé Collision 11](#_Toc124344794)

[2.17 Présentation détaillé Camera 11](#_Toc124344795)

[2.18 Présentation détaillé Chrono 11](#_Toc124344796)

[2.19 Présentation détaillé Life 11](#_Toc124344797)

[3 Conception graphique 11](#_Toc124344798)

[4 Partie Algorithmie – Intelligence artificielle 11](#_Toc124344799)

[4.1 Explications 12](#_Toc124344800)

[4.2 Extrait de code 12](#_Toc124344801)

[5 Cahier de recettes 13](#_Toc124344802)

[5.1 Tests de validation 13](#_Toc124344803)

[5.2 Tests de performance 13](#_Toc124344804)

Attention : toute vos impressions écrans doivent être lisibles !

# Présentation

## Description générale

## 

Man-chaud est un jeu de plateforme dans lequel l’utilisateur incarne un pingouin. L’objectif est de récupérer les différents morceaux de portail disséminer dans la map afin qu’il puisse revenir à sa banquise. Chaque niveau du jeu se trouve sur une map différente.

*Genre et principe du jeu, but du joueur.*

*Impressions écran, avec des phrases introductives et explicatives.*

## Règles du jeu

Durant la partie, le joueur est susceptible de rencontrer différent prédateur. Le pingouin pourra se défendre mais s’il se fait toucher il perdra une de ses vies. Lorsque le pingouin perd ses trois vies ou tombe dans le vide, il meurt et la partie se termine.



Barre de vie

Contrôle :

* Afficher le menu : Tab

Les déplacements :

Le pingouin peut marcher, a droite ou a gauche avec les flèches respective.

Pour sauter, touche espace. Attention, le pingouin ne peut sauter que s’il a un contact avec le sol.

Pour glisser, la flèche du bas. Ainsi, le pingouin se déplacera plus vite.

Le pingouin peut attaquer ses ennemis en leur sautant dessus ou avec la touche « entrer », lui faisant lancer une boule de neige devant lui.

Les éléments récoltables :

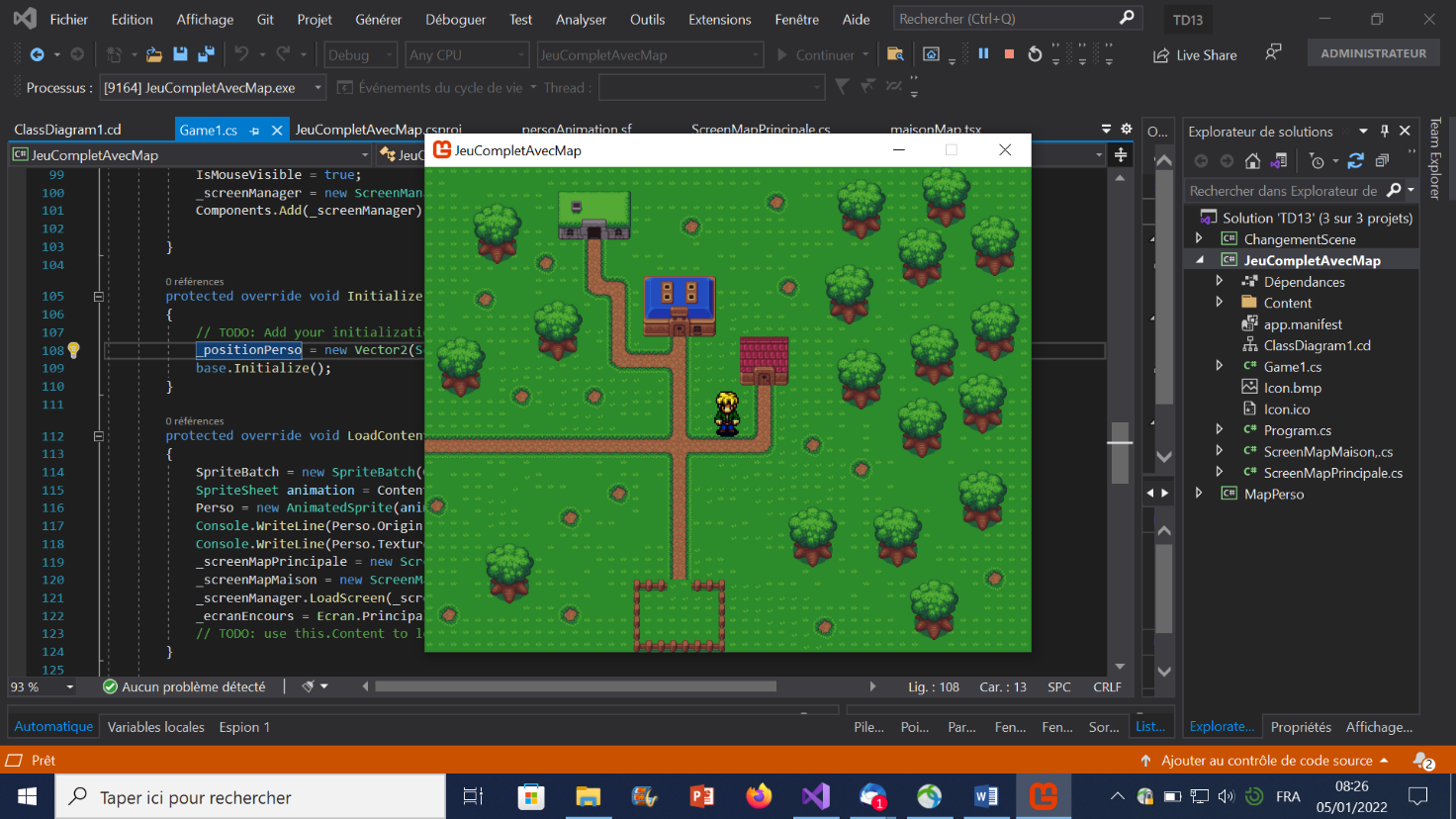
* Les pièces : Attention, si les pièces peuvent redonner de la vie lorsque le pingouin en a perdu, celles-ci peuvent également le faire ralentir par leur poids !
* Les morceaux de portail : Afin de remporter le niveau, il est nécessaire de tous les récolter. Leur nombre dépend du niveau, il est affiché sous la barre de vie, en haut à gauche de l’écran.

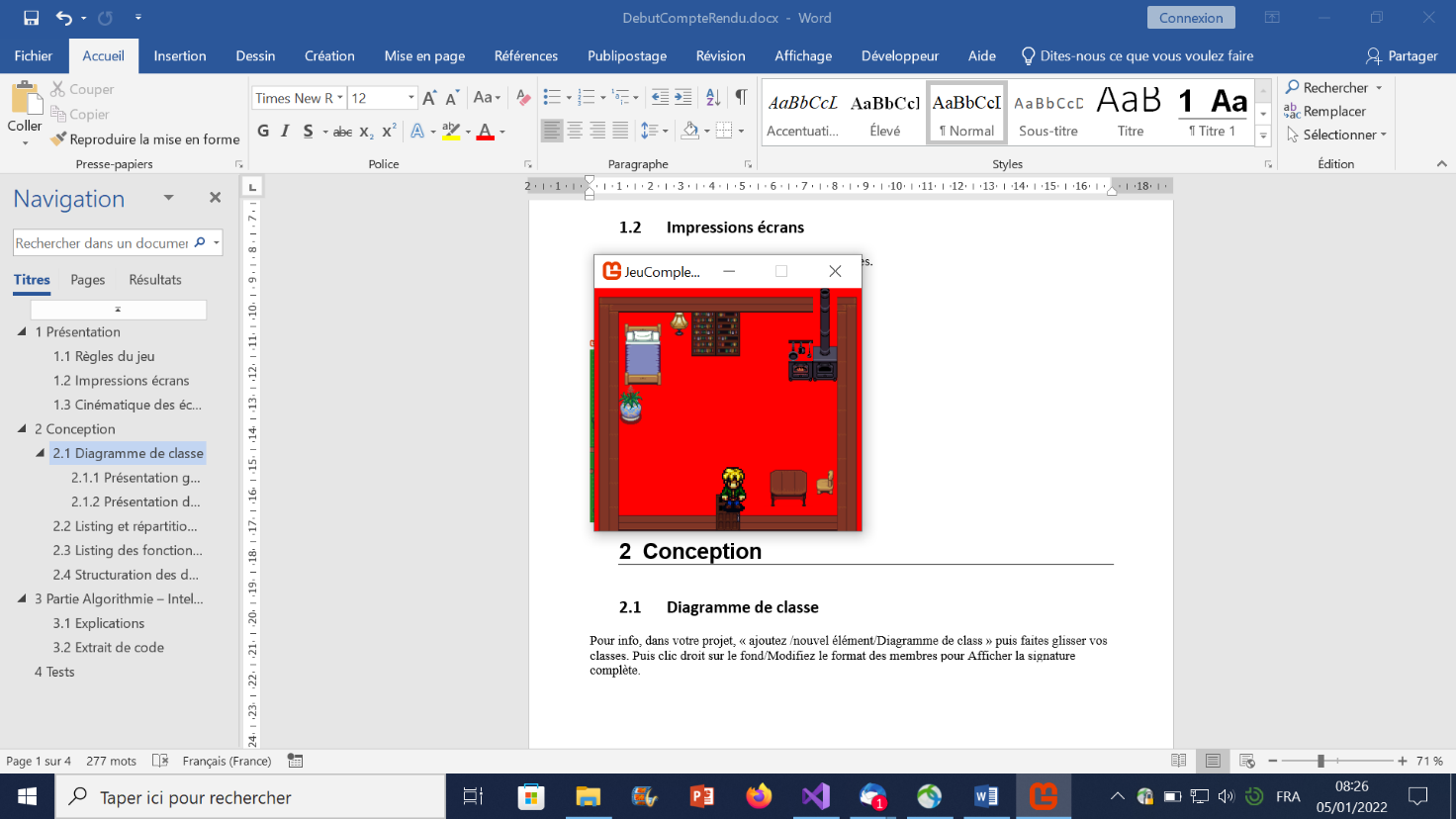
Codes triche :

* Pour les activer : F2
* Pour rendre toute sa vie au pingouin : V
* Pour voler et traverser les murs au-dessus du pingouin : F
* Pour avoir tous les morceau de portail : C
* Pour se téléporter au point de départ : Inser
* Pour se téléporter a la fin de la map ou le portail de fin s’ouvre : Fin
* Pour remettre le pingouin a sa vitesse initiale : P

Règle du jeu détaillée ainsi que la description des touches ou autre nécessaire pour jouer (code triche par exemple …)

## Cinématique des écrans





Arrivé à la porte bleue



Clic sur Notes de pingouin



Clic sur charger un niveau

# Conception – Diagramme de classe

## Présentation générale

*Faites une impression écran générale (sans le détail des classes ) accompagnée d’explications synthétiques pour chaque classe. N’héitez pas à regouper et découper si schéma trop grand.*

*Aide : sur votre projet, « Ajoutez un nouvel élément / Diagramme de classes » puis faites glisser vos classes depuis l’explorateur de solution dans la fenêtre*

*Justifiez vos choix de conception : expliquez et mettez en évidence si vous avez fait des classes pour améliorer, factoriser votre code .(Ex : classe Sprite ou Personnage, ….)*

*Ex :*

*Game1 est le point d’entrée : elle contrôle tous les enchainements des différents écrans ainsi que les pauses ou sortie du jeu.*

****

****

## Présentation détaillée Game1

Puis faites une impression écran détaillée de chaque classe avec leur signatures accompagnée d’explications un peu plus détaillées.



Game1 : instancie les 7 GameScreen : ChoixNiveau, Desert, GameOver, Menu, Regle, Snow et Win.

Game1 charge en tout 1er Menu, puis selon le clic l’utilisateur les booléen changent permettant à Game1 de charger Regle, ChoixNiveau, ou Desert. Si Game1 charge ChoixNiveau, il peut alors de nouveau charger Menu, ou charger Desert ou Snow.

Elle contient 28 champs :

* \_choixNiveau : c’est un objet de classe ChoixNiveau qui permet à Game1 de charger le GameScreen ChoixNiveau.
* \_desert : c’est un objet de classe Desert qui permet à Game1 de charger le GameScreen Desert.
* \_gameOver : c’est un objet de classe GameOver qui permet à Game1 de charger le GameScreen GameOver.
* \_graphics : c’est un objet de classe GraphicsDeviceManager
* \_menu : c’est un objet de classe Menu qui permet a Game1 de charger Menu.
* \_regle : c’est un objet de classe Regle qui permet à Game1 de charger le GameScreen Regle.
* \_screenManager : c’est un objet de classe ScreenManager qui sert à gérer le chargement des GameScreen ….
* \_snow : c’est un objet de classe Snow qui permet à Game1 de charger le GameScreen Regle.
* \_win : c’est un objet de classe Win qui permet à Game1 de charger le GameScreen Regle.
* Attaquer : c’est un objet de classe Keys
* clicMenu : c’est un objet de classe bool
* clicWin : c’est un objet de classe bool
* dernierePositionPingouin : c’est un objet de classe Vector2 qui sert a garder en mémoire dans game1 la position du pingouin durant la parti
* droite : c’est un objet de classe Keys, cette variable permet de stocker la touche de déplacement vers la droite
* gauche : c’est un objet de classe Keys, cette variable permet de stocker la touche de déplacement vers la gauche
* glisser : c’est un objet de classe Keys, cette variable permet de stocker la touche de glissade
* sauter : c’est un objet de classe Keys, permet de stocker la touche de saut
* goDead : c’est un objet de classe bool qui permet de gérer si Game1 doit charger le gameScreen GameOver.
* goDesert : c’est un objet de classe bool qui permet de gérer si Game1 doit charger les GameScreen Desert.
* goRules : c’est un objet de classe bool qui permet de gérer si Game1 doit charger les GameScreen Rules.
* goSnow : c’est un objet de classe bool qui permet de gérer si Game1 doit charger les GameScreen Snow.
* goStop : c’est un objet de classe bool qui permet de gérer si Game1 doit fermer la fenêtre.
* HAUTEUR\_FENETRE : c’est un objet de classe int qui permet de définir la hauteur de la fenêtre.
* LARGEUR\_FENETRE : c’est un objet de classe int qui permet de définir la largeur de la fenêtre.
* nivActu : c’est un objet de classe int qui permet de savoir la parti ou en est l’utilisateur.
* pause : c’est un objet de classe bool qui permet de gérer si la partie en cour est en pause.
* police : c’est un objet de classe SpriteFont qui permet de globaliser une police pour toute les classe.
* reprendre : c’est un objet de classe bool qui permet de gérer si une partie est a reprendre.

## Présentation détaillé Menu



Menu :

Elle contient 15 champs :

* \_mouseState : c’est un objet de classe MouseState qui permet de savoir l’etat de la souris.
* \_myGame : c’est un objet de classe Game1 qui permet d’heriter des fontion de Game1.
* \_pingouin : c’est un objet de classe Pingouin qui permet d’utiliser la classe Pingouin.
* \_pingouinG : c’est un objet de classe Pingouin qui permet d’utiliser la classe Pingouin.
* \_positionFond : c’est un objet de classe Vector2 qui permet de donner une position au fond.
* \_textureFond : c’est un objet de classe Texture2D qui permet de charger une texture.
* clicChoixNiv : c’est un objet de classe bool qui permet de renvoyer a Game1 si l’utilisateur clic sur Charger un niveau.
* jouer : c’est un objet de classe string qui permet d’initialiser le texte a afficher.
* niv : c’est un objet de classe string qui permet d’initialiser le texte a afficher.
* positionJouer : c’est un objet de classe Vector2 qui permet de donner une position au string jouer.
* positionNiv : c’est un objet de classe Vector2 qui permet de donner une position au string niv.
* positionQuitter : c’est un objet de classe Vector2 qui permet de donner une position au string quitter.
* positionRegle : c’est un objet de classe Vector2 qui permet de donner une position au string regle.
* quitter : c’est un objet de classe string qui permet d’initialiser le texte a afficher.
* regle : c’est un objet de classe string qui permet d’initialiser le texte a afficher.

## Présentation détaillé ChoixNiveau

ChoixNiveau :

Elle contient 9 champs :

* \_messMenu : c’est un objet de classe string qui permet d’initialiser le texte a afficher.
* \_mousState : c’est un objet de classe MouseState qui permet de prendre
* \_myGame :
* \_niv1 :
* \_niv2 :
* \_positionMenu :
* \_positionNiv1 :
* \_positionNiv2 :
* \_textureBackground :

## Présentation détaillé Regle

## Présentation détaillé Desert

Une image contenant texte

Description générée automatiquement

Desert est un niveau à part entière, c’est le 1er. Il s’occupe de ce qui se passe dans cette scène. Toutes les instances des pièges, récompenses, ennemies, boule de neige, parties de portal sont regroupés sous forme de tableau ou de liste dans le but de pouvoir en ajouter facilement à notre guise.

\_fondDesert est l’image de fond du niveau.

\_cheat est un boolean qui stock s’ils sont activés ou non.

\_heartSprite conserve le sprite du cœur et le tableau de position de cœur permet d’en afficher le nombre souhaité.

\_closingPortal

\_openingPortal

\_partiesRecoltees stock le nombre de partie de portail récupérée.

\_keyboardState permet d’utiliser le clavier dans le jeu.

\_manager est une instance de GameManager.

\_pingouin est une instance de Pingouin.

\_tiledMap et tiledMapRender servent à utiliser et afficher la map correctement.

// à enlever – variables non utilisées

Les rFox etc…

hauteurRecompense1

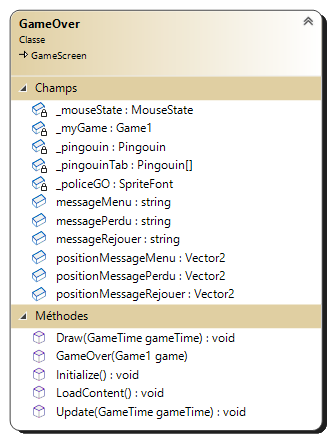
scale

## Présentation détaillé Snow

Inserer image class snow

Desert est le 2ème niveau. Il s’occupe de ce qui se passe dans cette scene.

## Présentation détaillé GameOver



## Présentation détaillé Win

## Présentation détaillé Pingouin

## Présentation détaillé Snowball

## Présentation détaillé MonstreVolant

## Présentation détaillé MonstreRampant

## Présentation détaillé Trap

## Présentation détaillé Recompenses

Recompenses :

Elle contient 8 champs :

* effetPiece :
* etat :
* hauteur :
* largeur :
* position :
* rectangleSprite :
* sprite :
* typeRecompense :

## Présentation détaillé Collision



## Présentation détaillé Camera

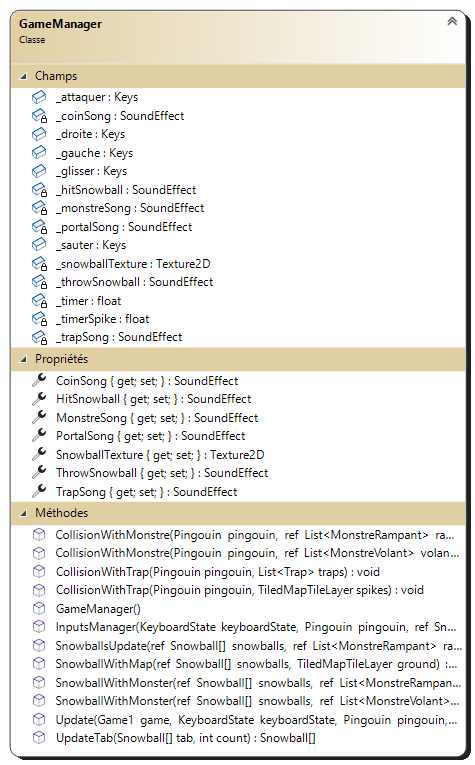


## 

## Présentation détaillé Chrono



## Présentation détaillée GameManager



# Conception graphique

Nous avons réalisé nos deux maps nous même, en utilisant des Tilesheets trouvées sur itch.io. Nous avons réalisé nos maps a l’aide de Tiled et avons effectuer plusieurs séries de tests afin de vérifier que le parcours était possible avec le pingouin.

Nos décors ont été trouvé eux aussi sur internet, tout comme nos sprites que nous avons par la suite animés sans modifier les originaux trouvé sur itch.io.

Indiquez si vos maps, décors,sons sont des sources existantes, dans ce cas donnez leur provenance Expliquez les retouches ou création que vous avez peut être réalisés.

# Partie Algorithmie – Intelligence artificielle

## Explications

Expliquez ici la logique et les algos choisis : mettez en évidence leur complexité.

Il est possible d’utiliser des algorithmes existants (ex : A\*) dans ce cas la expliquer la façon dont vous les avez utilisés/codés.

## Extrait de code

Mettez ici des extraits de code commenté de la logique précédemment décrite.

# Cahier de recettes

## 5.1 Tests de validation

|  |  |  |
| --- | --- | --- |
| Nom | Fonctionnalité | Etat |
| Gruson | Sprite Ennemi | Non achevé/ Bug/ OK |
| Labauve | Menu principal | OK |
| Labauve | Sprite Coin | OK |
| Labauve | TilesMap desert | OK |
| Sauthier | Classe Pingouin | OK |
| Clerc-Renaud | TilesMap snowmap1 | OK |
| Clerc-Renaud | Trap | OK |
| Clerc-Renaud | Sprite trap | OK |
| Clerc-Renaud | Sprite renard | OK |
| Sauthier | Sprite pingouin | Ok |
| Clerc-Renaud | Sprite eagle | OK |
| Labauve | Sprite portal | OK |
| Labauve | Recompenses | OK |
| Clerc-Renaud | MonstreRampant | OK |
| Clerc-Renaud | MonstreVolant | OK |
| Labauve | GameOver | OK |
| Labauve | Regle | OK |
| Labauve | Win | Non achevé |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 5.2 Tests de performance

A l’aide des outils de diagnostiques : vous prendrez quelques mesures (captures écrans) à des moments clefs de votre jeu de l’utilisation de la mémoire et du processeur . Vous commenterez bien évidemment les moments choisis et les mesures.

